(19) Europäisches Patentamt
European Patent Offi
Office uropé n d brev ts

(11) Publication number: 0 661 631 A2

(12) EUROPEAN PATENT APPLICATION

(21) Application number: 94850227.3

(51) Int. Cl.[6]: G06F 9/44

(22) Date of filing: 19.12.94

(30) Priority: 23.12.93 US 172290

(43) Date of publication of application:
05.07.95 Bulletin 95/27

(84) Designated Contracting States:
DE FR GB IT NL SE

(71) Applicant: TELEFONAKTIEBOLAGET LM ERICSSON
S-126 25 Stockholm (SE)

(72) Inventor: Pruitt, Leonard
5125 Hardaway Circle
The Colony, Texas 75056 (US)

(74) Representative: Lövgren, Tage et al
Telefonaktiebolaget L M Ericsson
Patent and Trademark Department
S-126 25 Stockholm (SE)

(54) Method and apparatus for creating a flowchart using a programmed computer which will automatically result in a structured program.

(57) The present invention relates to a method of creating a structured flowchart using a programmable computer and a programmable computer display such that any program created from the flowchart would be a structured program, the method including the steps of displaying a predetermined set of basic flow forms on a first display area, providing a means by which a user can select two flow forms from the set of flow forms, combining the two selected flow forms by placing one of the flow forms inside any statement box of another of the selected flow forms to yield a new valid flow form according to information provided by the user, and displaying the selected flow form and any new valid flow forms in a second area on the display. An apparatus for assembling a flowchart on a programmable computer display is also provided.

EP 0 661 631 A2

BACKGROUND

The present invention is directed to a method and apparatus for creating a structured computer program, that is, a method and apparatus for creating a flowchart using a programmed computer which will automatically result in a structured program. In particular, the present invention relates to a method and apparatus for creating a flowchart such that any program code created from the flowchart will be a structured program.

There are numerous existing flowchart programs (see, for example, U.S. Patent Nos. 4,546,435, 4,872,167, 4,852,047); however, none of the existing systems encourage, promote or require that the flowcharts which they create are structured according to the rigorous graphical notation used in the present invention. Source code or flowcharts which are not structured are said to be a "can of worms" or "spaghetti". There are degrees of "spaghetti" depending on the degree of deviation from pure structured flowchart or code. Non-structured flowcharts (or code derived from non-structured flowcharts) are known to have more errors and be more difficult to maintain than structured flowcharts or code.

Companies that deliver products with embedded computers (such as the products that telecommunication switching-equipment companies deliver) depend more and more heavily on software. This is because the value added to the product, the software included, is becoming a larger and larger part of the investment in the product. The intelligence provided by the software is more and more valuable to the customer.

Customers want products delivered ever faster. They want more intelligence in the products they buy. For systems such as, for example, telephone switches, this means more features in the switch. The features requested and desired become more complicated each year. However, companies have often had great difficulty in delivering software with the desired functionality on time and with high quality. The company able to deliver high-quality software on time, with the desired functionality with have a tremendous competitive advantage over a company that is not able to do this.

The larger the quantity of code in a product or project, the more important it is that the code be structured. Structured code is also a very important component of the method to use to develop correct code. Code that is correct upon delivery is much less expensive overall to create than code that must be corrected after delivery. The software designer is perhaps the most scarce and expensive resource in this process of code development. The designer's time must be used in an effective way. The most effective way for a software designer to use his time is to design and write structured code. This goal is difficult to achieve using today's software development tools, since it is left to the designer to ensure that the flowchart or code he is developing is structured.

SUMMARY

It is accordingly an object of the present invention to provide a software development tool that will enable software designers to easily and efficiently develop structured software so as to achieve a better quality product, made less expensively.

It is another object of the present invention to provide a software development tool that will enable the development of software that is structured to enable the delivery of complex, software-intensive products on time and with higher quality.

It is another object of the present invention to provide a software development tool that will enable the software developed to have the properties necessary to help ensure profitability.

According to one embodiment of the present invention, a method of creating a flowchart using a programmable computer and a programmable computer display, such that any program created from the flowchart would be a structured program, the method including the steps of displaying a predetermined set of basic flow forms on a first display area, providing a means by which a user can select two flow forms from the set of flow forms, combining the two selected flow forms by placing one of the flow forms inside any statement box of another of the selected flow forms to yield a new valid flow form according to information provided by the user, and displaying the selected flow form and any new valid flow forms in a second area on the display. The basic flow forms are a statement box, a sequence of two statement boxes, iteration forms for FOR and WHILE statements, and alternation forms including boolean forms for IF-THEN and IF-THEN-ELSE statements and CASE statements.

According to an embodiment of the present invention, an apparatus is provided for assembling a flowchart on a programmable computer display including means for displaying on the screen a set of basic flow form icons, each icon comprising at least one polygon having one input line and one output line, each icon having one input line and one output line, user input means for enabling a user to select at least one of the basic flow form icons, and means for displaying the selected icon on a different area of the screen in a position designated by the user.

The apparatus according to one embodiment of the present invention further includes user input means for inputting text to be associated with the at least one polygon in the at least one selected icon, and means for displaying the inputted text in an area adjacent the at least polygon in the at least one selected icon.

Each icon in the apparatus according to one em-

b dim nt of the pres nt invention includes a rectangular or square polygon and th m ans for displaying th selected icon includes means for combining two s l cted flow form icons, means for determining wh n th position designated for displaying the selected icon by the user is correct according to predetermined rules, and means for prohibiting operation of the means for displaying the selected icon responsive to a determination by the means for determining that the position designated is incorrect according to the predetermined rules.

According to an embodiment of the present invention, an apparatus is provided for assembling a flowchart on a computer display including means for displaying in a first area of the display a predetermined plurality of flowchart icons, each icon comprising at least one statement box, an input line and an output line, user input means for indicating selection of one of the plurality of flowchart icons, user input means for indicating placement of the selected icon in a second area of the display, means for determining whether the indicated placement of the selected flowchart icon satisfies predetermined rules, the rules allowing for placement of one icon inside a statement box of another previously selected icon to form a new flowchart icon, allowing for placement of one icon in a head-to-tail relationship with another icon, allowing only downward flow of control within the flowchart, and prohibiting crossing of any flow line within the flowchart, and means for temporarily adding to the predetermined plurality of flowchart icons any new flowchart icon created by the user input means for indicating placement.

Still other objects, features and attendant advantages of the present invention will become apparent to those skilled in the art from a reading of the following detailed description of the embodiments constructed in accordance therewith, taken in conjunction with the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention of the present application will now be described in more detail with reference to the preferred embodiments of the device, given only by way of example, and with reference to the accompanying drawings, in which:

Fig. 1 is a schematic illustration of a computer system in which the present invention is implemented;

Figs. 2A-2F illustrate the basic flow form icons used in a preferred embodiment of the present invention;

Figs. 3A-3E illustrate various flowcharts that can be created according to a preferr d mbodiment of th present invention;

Figs. 4A-4E illustrate flowcharts of a pref rr d embodiment of th present inv ntion; and

Fig. 5 is an exemplary schematic illustration of a display on which the us r int rfac according to a pr f rred embodiment of the present inv ntion is display d.

DETAILED DESCRIPTION

In this description, the word "structured" is used as an adjective to describe code or flowcharts which have particular and very precisely defined properties. When it is used as in the phrase "please structure this piece of code", it is a verb which indicates a particular action which is to be performed on the code or flowchart.

All code which will compile without errors or assemble without errors (in the case of assembly language) is "structured" in the sense that it is "organized" at some level. If the code were not organized in the sense of meeting the syntactic rules of the language, then it would not compile or assemble. However, that is not at all what is meant by the word "structured" in this application. The word "structured" as it is used herein has a precise definition which will be described further below.

The program according to the present invention, which has the novel attribute that it can require structured flowcharts will, by virtue of the inherent properties of structured flowcharts, increase the probability that the flowcharts, created by the use of it, will be more nearly correct. Flowcharts or code derived from them which are correct from the beginning are very economically advantageous. The present invention may save millions of dollars per year for those who employ it.

Figure 1 schematically illustrates an embodiment of the hardware which can be used to implement the present invention. A CPU 10 is provided which executes the program code for implementing the present invention, which is stored in ROM 30. There is also RAM 35 for storing temporary data. A display 15 is provided connected to the CPU on which the user interface of flowchart icons is displayed along with the display area in which the flowchart itself is created by the user. Keyboard 20 is provided for the user to input the commands to the computer program. A mouse 25 may also be provided to allow additional user control, such as click and drag movement of the flowchart icons provided according to the present invention.

There are six basic flow forms or flow form icons that may be combined according to a set of rules to produce structured flowcharts according to a preferred embodiment of the present invention. These six basic flow forms are shown in Figures 2A through 2F discussed below. For purposes of this discussion, th t rms "flow forms" and "flow form icons" will be used interchangeably.

According to th present invention, thre rul s should be followed in combination with th use of th

basic flow forms to forc a structured program as follows:

1. Sel ct two flow form icons from the set of valid flow form icons. Choosing the same flow form icon twic counts as selecting two flow form icons. Those two selections must be combined according to rule 2.
2. The combinatorial rule: any valid flow form may be placed inside any statement box of another valid flow form. It is permissible to place a copy of a flow form icon inside one of its own statement boxes.
3. Any flow form which has been derived according to the combinatorial rule is itself a valid flow form icon and may then be used as if it were one of the six basic flow form icons. Program development continues (if necessary) by returning to step 1 and selecting among the set (now possibly extended) of valid flow form icons.

By definition, then, a structured flowchart according to the present invention is one which was (or could have been) created using the six basic flow form icons and following the above three rules. As a practical matter, the implementation of the program according to the present invention includes an exception to the second rule which allows head-to-tail connection of two selected flow forms as opposed to inserting one into a statement box of another. This is allowed because the implementation of the program according to the present invention is not required to work exactly according to the three rules. What it must do is yield results that could have been obtained by following the three rules. For example, to create a simple program with just three sequential statements, one could choose a sequence flow form (Figure 2B) and choose another sequence form and place it inside the second statement box of the first sequence flow form. When the combination formed is expanded, it results in a three statement sequence flow form. As a practical matter, rather than requiring the user of the present invention to choose that sequence of flow forms, one embodiment of the present invention (as shown in Figure 4D) allows the user to choose a single statement icon, followed by a sequence icon placed in a head-to-tail relationship to yield the same three statement sequence flow form.

There are a number of rules and flow form icons or graphs which could be developed to implement and create a structured program. The set of rules and flow form icons that are listed above is not the only possible set.

In trying to establish a specific set of rules and flow forms, there is a balance which must be struck between the number of flow forms and the number and complexity of th rules. According to th present invention, a system is being stablish d that can b used to cr at any program whatso ver that can b written. This means that the rul s have to be able to cover programs that ar one line long as well as th s that are 100 million lines long. In ord r to cov r this rang , th stat ment icon of Fig. 2A and sequ nce icon of Fig. 2B ar both included in the s t of valid flow forms according to a preferred embodiment of th present invention.

For example, if a designer wishes to write a program which is exactly one line long and wishes to do it according to the rules of structured programming according to the present invention, then the single statement flow form will be chosen twice and inserted into itself. Then the process stops. The program is written. It is structured because it has been written according to the rules.

If either or both of the first two flow form icons were omitted, then the rules would have to become more complicated in order to specify, according to an algorithm, how to structure code. Following these rules guarantees that the program that is produced will be structured.

Within the scope of the invention, it is also possible to define a slightly different set of flow forms and a slightly different set of rules and get the same results. The rules, as they are described above, allow the user to write a program (or subroutine) only one line long. (The user chooses the statement box flow form and puts it inside itself.) However, a special rule may be added that requires that if only one statement (flow form) is needed in the program, then the user can select the one desired flow form and then stop. The cost of this is the addition of the new rule, since there are still six basic flow forms and an additional rule.

The rules given above are complete. If a program has been or could have been, constructed according to the three rules for structured coding in conjunction with the six basic flow form icons, then that program is structured. If a program (or flowchart) has been devised in such a way that it is not possible to re-create it by following the rules for structured code, then the program is not structured.

Thus the standard for determining whether or not a program is structured is absolutely objective. No subjective judgement is required. Thus, a computer program according to the present invention is ideal for determining whether the program is structured.

The six basic flow forms will now be discussed with reference to Figures 2A-2F. As the flow forms are displayed using the user interface portion of the program according to a preferred embodiment of the present invention, they are also referred to as icons.

Figure 2A illustrates a statement icon and Figure 2B illustrates a sequence icon according to a preferred embodiment of th present invention. Figure 2C represents iteration forms for indicating FOR and WHILE language constructs according to a pr ferred embodiment of th present invention. Figur s 2D and 2E illustrate one type of alt rnation form icons, and

in particular boolean form icons, representing th IF-THEN and IF-THEN-ELSE languag constructs according to a preferr d mbodiment of the pres nt invention. Figure 2F illustrates another type of alternation form icon r presenting th CASE languag construct according to a preferred embodiment of the present invention. That is, depending on the results of the test specified in or adjacent to the diamond, one of the paths is chosen from the set of paths in the flow form.

Each icon consists of one or more polygons connected by input line I and an output line O. Each icon has only one input line I and one output line O. Each polygon in the icon also has only one input line and one output line. The output line for each icon is placed directly below the input line for that icon.

The FOR and WHILE statements both require a boolean expression within them to function; however, since the emphasis within these forms is on looping rather than branching, they are referred to as iteration forms rather than boolean forms.

A number of conventions are implemented in the program according to the present invention relating to the boolean flow forms, the IF-THEN and IF-THEN-ELSE boolean statements. In particular, the IF-THEN offers two paths. The TRUE side has a statement box whereas the FALSE side does not have a statement box. This rule is built into the icon by the provision of the TRUE and FALSE labels within the icon. By definition, the IF-THEN does not have a statement box in the FALSE path. This convention has been established according to the present invention for flowcharts because this is a convention for source code. Part of the goal is to make the flowchart and source code alike. The IF-THEN statement in source code cannot, by definition, have a statement box for the FALSE path.

If the user wishes to execute a statement box in the IF-THEN statement when the boolean expression is false, then a NOT should be put in front of the boolean expression. That will negate the expression. If the expression then evaluates to false, the NOT will make it true, and so the statement box of the IF-THEN will be executed when the inner boolean expression is false. Note that the boolean expression as a whole (including the NOT) should be true when the statement box is executed.

As can be seen, the TRUE and FALSE alternative paths are not drawn coming out of the sides or bottom of the decision diamond as was conventional practice in the past. This is because it is often difficult to place the complete boolean expression inside the decision diamond. There is plenty of room to place even a long boolean expression beside the d cision diamond; however, this requires movem nt of th paths to just below th diamond itself.

According to th present invention an IF-THEN or IF-THEN-ELSE is only us d in conjunction with a boolean expression. The bool an expression can only valuat to TRUE or FALSE. There are no alternativ s.

By convention according to th pr sent inv ntion, the TRUE branch is plac d on the right sid for both the IF-THEN and the IF-THEN-ELSE statements. This convention is based on the fact that virtually all languages are defined in such a manner that for IF-THEN-ELSE statements, if the boolean expression evaluates to TRUE, the first or top set of statements is executed. If the boolean expression evaluates to FALSE, then the second or bottom set of statements is executed. According to Western culture, top is associated with right is associated with true. Similarly, bottom is associated with left is associated with false. Thus, the drawing convention according to the present invention is an extension of architectural conventions of computer languages in combination with standard cultural traditions.

In Figures 2D and 2E, the true-false labels are provided in the icon according to a preferred embodiment of the present invention. The "IF boolean" expression is not; it is user-added using the program according to a preferred embodiment.

In the structured flowcharting technique implemented by the present invention, there can be no lines which flow upward. Since there are no lines that flow upward, that means that there is never any need to use arrows in the flow lines. All flow lines flow downwards. This convention reduces clutter in flowcharts. Since they are then cleaner, they are also easier to read. There is never a need to look for arrows to indicate direction of flow. Another significant convention is that there is never any need for flow lines to cross one another in a structured flowchart.

In principle, off page connectors are not allowed in flowcharts developed according with the use of the method and apparatus according to the present invention. There are a number of exceptions to this rule. Off page connectors are allowed in the case of the CASE statement that has such a large number of cases that the CASE statement runs off the page.

When the flowchart is drawn describing a finite state machine (FSM), the state symbol behaves like a CASE statement. Each signal that can be received within that state is analogous to a case in a CASE statement. A state of this type is also called a "wait" state. The wait state is effectively acting as a signal reception point. Off page connectors are allowed in the case of a state that has a large number of possible signals or events that can be received at that point. It is considered within the skill of the ordinary artisan to use off page connectors in the implementation of the program according to a pr ferred embodiment of the pres nt invention.

Figure 3A illustrates how a flow form icon can b placed inside an appropriat statement box of another flow form icon. Th resulting n wly constructed valid

flow form is shown in Figur 3B, which illustrates an IF-THEN-ELSE flow form with an IF-THEN flow form inside the stat m nt box on its TRUE side. This newly constructed valid flow form can then b listed among th original six basic flow forms icons. At this point, the newly constructed flow form may be combined with another flow form by placing the new flow form inside one of the other's statement boxes or some pre-existing flow form may be placed inside one of the new flow form statement boxes. Also, a newly constructed flow form can be duplicated and placed inside one of its own statement boxes. According to the present invention, these constructed flow forms are temporarily stored in memory and do not become part of the predetermined set of six basic icons.

This process of combining valid flow forms is repeated continuously by the user until the program he wants is finished.

Figure 3C shows an IF-THEN icon placed inside a FOR loop. Inside the IF-THEN statement there is a sequence of two statements, shown in Figure 3C as a statement box and a signal sending statement box. The signal sending statement box is an example of an individual, rectangular statement box changed to a special shape to represent a special type of statement. These special shapes may be implemented as the shapes typically used in known flowcharting techniques for statements such as sending output to a printer device, storing a value in memory, etc. These special shapes do not affect the flow through a program, but are included to conform the program to requirements of the application. The choice of which of these special shapes are available is left to the ordinary artisan in possession of this disclosure. One embodiment of the present invention, not shown in the drawings, is programmed such that after the user chooses one of the basic flow forms, he can then change a rectangular polygon in the form to one of the available special shapes.

Figure 3D illustrates how CASE statements often must be handled. Very often the contents of each case must be reduced to a subroutine call in order to enable clear understanding of the functioning of the statement. By the same token, the CASE statement itself must very often be placed inside a routine as the only statement in the body of the routine. This is quite often necessary in order to allow the code associated with a CASE statement and other closely related statements, outside the CASE statement, to fit together on one page.

When flowcharts are created according to the method described above, the flowchart results have certain emergent properties that are characteristic as illustrated in Figure 3E.

First, the ntry point at the top of the modul of cod drawn is always xactly abov th xit point at th bottom. Further, it is also possible to draw a box around each area that used to be a statem nt box. Finally, if a box is drawn ar und an area that used to be a stat m nt box, that box will be inters ct d by only two lines, on at th top center and on dir ctly be- n ath th top lin at bottom center.

With respect t th case statement of Figur 2F, it is shown having only three options or three alternative paths. In actual practice, case statement may have two or more alternative paths. According to a preferred embodiment of the present invention, the user is prompted to allow him to specify the number of paths desired in the icon.

As mentioned above, programs written using the flowchart according to the present invention require that the flowchart for any particular routine must be contained on one page. This is true except for the case situation in which a case statement includes a list of alternatives which is too long to be shown on only one page. To accomplish this, designers should start at the top and work down layer by layer, one at a time, until the final level of detail is reached.

In order to satisfy the prohibition against page breaks, and thereby to create modular code, it is necessary to break a program down into levels. Each level should be a module which is called by the preceding, or some other, level.

The basic rule for creating a level is that from the higher level, one should be able to see the outline of or references to the next lower level. For example, in the design of a ship, one can, in principle, see from the drawing of the ship as a whole, the set of decks of which the ship is comprised. Another example is that of the drawing of a building. From the drawing of the building as a whole, one can, in principle, see approximately how many floors there are in the building.

From the drawing of a specific deck on a ship, one can see where each of the rooms on that deck will be. Likewise, from the drawing of a specific floor plan within a building, one can see each of the rooms on that floor. From the drawing of the engine room on a ship, one can see the engine. From the drawing of an engine, one can see a piston.

This is a rule for design that has been worked out in practice over centuries of engineering. The basic principle of hierarchies described here has been around for much longer than that, however. Engineers are simply imitating nature in following the practice of dividing systems into subsystems and then repeating this process the necessary number of times.

It is absolutely commonplace to find an IF-THEN statement in a program (or flowchart) to be scattered across multiple sheets of paper or screens. A given subroutine (assuming that subroutines even exist in the program) may be scattered across a number of pages. This situation is so commonplace that most designers do not v n think of it as being wrong. It is assumed that it is a necessary part of th design of large programs. It is in fact absolutely not nec ssary. Not only is it not nec ssary, it is extremely d struc-

tiv .

Programs should be d signed as other obj cts of engin ering practic . There should b one page that describes th program as a whole. For programs that are a single proc ss and th process is not represented by a finite state machine this is virtually always possible to do.

For software units where there are multiple processes, the rule can be modified to say that each process should be described on one page. If the process is modeled as a finite state machine, then it is likely that even a high-level description (which actually makes some sense and does something) will not fit onto one page. In this case, each state of the finite state machine can be fitted onto one page (with occasional exceptions allowed there also). The page-per-module rule can then be applied very rigidly after the state/event intersection occurs. At that point, the software becomes completely identical with sequential (non-real-time) code from the point of view of engineering drawing approach.

Whether this is source code or flowchart does not matter. This is how the principle of hierarchies or layers should be applied to software.

This whole idea often comes as a total shock when proposed to a software designer for the first time. There is no reason, however, why software designers should have to fight with logical statements or signals or routines (in flow-chart or source-code form) that extend across multiple pages. All of the six basic flow forms should (with some exceptions for the CASE statement) always appear on one page. An IF-THEN or IF-THEN-ELSE or WHILE or FOR statement should always start on one page and end on that same page.

Following this rule alone will result in an immediate increase in the quality of software. It is a rule of thumb in virtually all other engineering disciplines except software engineering. Following the "one-page rule" will automatically cause software designers to think in terms of layers, levels, or hierarchies. It will also automatically put a great deal of pressure to represent the thing as a whole at the top level and put lower levels of detail at lower levels. That such a practice, which is absolutely standard in all other engineering disciples, and yet is almost unheard of in software engineering, is astonishing.

A module of code should be no more than one page long. When it becomes more than one page long, it should be broken down into routines.

Whatever can be put into a statement box (according to the rules for structuring) can be put into a routine. If the fragment of code cannot be surrounded by a statement box, then the fragm nt of cod is not a candidate for being convert d into a routin .

Generally, th r is no consid ration given in either flowcharts or source cod to considering the program as a set of drawings. Moreover, there is g nerally little or no consideration giv n to considering a program as a s t of modules. A modul as us d herein is a subordinate piec of a program as a whol , such as proc ss, routin , subroutin , or procedur .) In th standard "stream" approach the program is treat d as a monolithic whole without a breakdown into levels.

The flow forms should never extend across more than one page even if they do contain a long CASE statement. If a CASE statement inside another flow form will cause that flow form to extend across more than one page, then the CASE statement should be placed inside a subroutine. The subroutine call that contains the CASE statement will only occupy one line of source code. The same principle also applies to flowcharts created according to the program of the present invention. This means that all executable code (except for individual CASE statements or lists of signal reception points, that may be longer than one page) must be fitted into modules which are no longer than one page.

Thus, the whole program is to be broken down into units that, with the exception of declarations, lists of signal reception points, and some long CASE statements, each fit onto one page. Thus the program becomes a set of short "programs". Each module is always easy to understand by itself. It must be easy to understand by itself because it is so short that it fits on one page. In practice, this means that a module will be no longer than approximately thirty-three to fifty lines of code. Analogous requirements are placed on the flowcharts created according to the present invention.

The first priority of the designer must be to write a program that is correct. All other considerations must be secondary to this top priority. In order for a designer to be able to write a correct program, there must be organization and order in the code. In order to successfully employ the engineering drawing principle, the designer must make extensive use of routines.

The primary means of providing order in a program is to use structured code. The second most powerful means of providing order in a program is to apply the engineering drawing principle to structured code so that the modules will always be short.

According to preferred embodiments of the present invention, the use of the rules and the icons described above can be implemented using a programmed computer of the type shown in Figure 1. Figs. 4A through Fig. 4B illustrate exemplary flowcharts of software routines which may be implemented in order to create the program according to the present invention. The coding of the program, the choice of the languag and the computer system to b us d is left to th discretion of the reader of this patent. Th principles of coding described above maybe implem nted in th program according to the present inv ntion as desir d using expert system and other programming

t chniques. It is c nsidered within the skill of the ordinary artisan to implement and cod th program according to the pr s nt inv ntion once in possessi n of th pres nt disclosur .

Figure 4A illustrates th main routine. St p S300 displays the predetermined set of flowchart icons in one area of the screen display. This display may consist of a windowing arrangement as currently used in many systems or a panel of predetermined icons, as shown in Fig. 5. Fig. 5 illustrates one exemplary display format that may serve as the user interface of the program according to an embodiment of the present invention. The area 15a includes the six basic flow forms icons shown in Figs. 2A-2F. Area 15b is used as the area in which the flowchart created by the user will be displayed.

Step S310 is a user input routine which is entered whenever it is detected that the user has pressed a key or moved the mouse.

Figure 4B illustrates the user input subroutine of step S310. A user input step S320 determines whether the user input is a selection of an icon. This selection may either occur by use of a mouse, cursor keys, or touch pad mechanism, e.g., from area 15a. If icon has been selected, the program follows the true path to step S330 which is a user placement routine. This will be described below with reference to Figure 4C. If the test in step S320 results in a false evaluation, it is tested at step S340 whether it is a text entry. If so, the TRUE path is taken and the user is prompted to input text associated with a previously selected icon. The text input by the user will be placed in the display area 15b adjacent the icon with which it is to be associated. This allows for greater readability, since the text may be larger than the polygon in the icon.

In conventional flowcharting programs, the user is given an opportunity to draw lines connecting various boxes. This is unnecessary in the program according to the present invention since the user placement routine allows icons to be placed either within a statement box of an existing icon or directly below an existing icon where the input line of the second icon is connected to the output line of the first icon. Thus, the flow lines, flowing only downward, are created automatically by the program according to the present invention based on the placement of the selected icon.

Figure 4C illustrates the user placement routine according to the preferred embodiment of the present invention. Step S360 determines whether the icon selected is the first icon to be placed in the display area 15b. If so, it is unnecessary to check for appropriate placement of the icon, so the routine ends. If not, the routine is ended. If so, the program nters the check rules routin at st p S370. At th nd of the check rules subroutine, at st p S380, it is d termined wh th r or not th flag is qual to on . This flag is us d by the check rul s routin to signal that an error has occurr d in the placem nt of the icon. If it is not qual to one, th rror m ssag routine is ent r d. If it is equal to on , the us r plac m nt routin nds.

Figure 4D illustrat s the ch ck rules routine. At step S395, a flag is qual to one. This flag signals whether or not the user has incorrectly placed an icon. Step S398 determines whether an icon has been moved to within a statement box. If so, at step S400, the newly created form is added to the available icons that the user may select and stored in temporary memory. These newly created forms may be displayed in a separate area of the display (not shown) so that the user can have access to them using the click and drag mouse, the cursor keys, or touchpad method.

If the answer at step S398 is false, it is assumed that the icon has been placed below the existing icons in the display. In this case, the input of the second icon is connected to the output of the first icon. At step S420, it is determined whether any flow other than downward control is necessary, such as if the icon has not been placed below the existing icons. If this is the case, at step S430, the flag is set to 2. If only directly downward control has been designated, it is determined at step S430 whether any of the lines of the different icons cross one another. If there is a crossing line, the flag is set, at step S440 equal to three. Then the routine ends. If no lines are found to be crossing, the routine ends.

At step S380 in Figure 4C, if the flag is equal to two or three, the error message routine is entered at step S390 and display of the selected icon is prohibited.

The error message routine is illustrated in Figure 4E. If the flag equals 2 at step S450, a particular error message is displayed indicating that only downward control is possible and the routine ends. If a flag equals three, at step S470, an error message is displayed indicating that the lines are crossing which is improper and the user is requested to re-enter the position of the new icon.

The basic features of a software implementation of a preferred embodiment of the present invention have been described herein. It is understood that the mechanism by which the icons are displayed, moved, and otherwise manipulated is within the skill of the ordinary artisan once in possession of the present disclosure.

An ideal situation for flowcharts and codes would be to have a computer-aided software engineering (CASE) tool available which would allow the designer to work in either source-code or flow-chart medium according to the preferences of the designer with the demands of the situation. The d signer could, for example, start out working in source cod to build up data types and other declarations and then switch to flowchart for xecutabl cod . As th designer would switch back and forth b tween flowchart and cod ,

th tool would automatically update the other form. In this manner, th flowchart and source code would automatically b maintained in parallel. This would ensur that th flowchart and code would b id ntical. Th goal of having the flowchart r pres nt th code at a "high level" could be achieved by having the code itself written in a hierarchial or layered manner. Therefore, although there would be flowcharts for even the smallest details of executable code, there would also be generated high-level flowcharts which would correspond to the higher layers of the source code. Even if such a tool were available (as far as is known to the inventor, no such practical tool exists today that meets exactly the requirements specified above), the present invention would still be useful in that it would help the user of the tool to extract maximum possible benefit from the CASE tool. Very few languages force structured code, neither do any of the flowcharting tools in existence force structured flowcharts ("structured" according to the definition herein). This means that even with modern tools and languages, it is still up to the designer to use self discipline to develop structured software products.

The foregoing description of the specific embodiments will so fully reveal the general nature of the invention that others can, by applying current knowledge, readily modify and/or adapt for various applications such specific embodiments without departing from the generic concept, and, therefore, such adaptations and modifications should and are intended to be comprehended within the meaning and range of equivalents of the disclosed embodiments. It is to be understood that the phraseology of terminology employed herein is for the purpose of description and not of limitation.

Claims

1. A method of creating a flowchart using a programmable computer and a display for the programmable computer, such that any program created from the flowchart would be a structured program, the method comprising the steps of:

   displaying a predetermined set of basic flow forms on a first display area;

   providing a means by which a user can select two flow forms from the set of flow forms;

   combining the two selected flow forms to yield a new valid flow form according to placement information provided by the user; and

   displaying the selected flow form and any new valid flow forms in a second area on the display.

2. The method according to claim 1, wherein th basic flow forms ar a statem nt box, a sequ nc of two statement boxes, iteration forms for FOR and WHILE statem nts, alternation f rms including boolean forms for IF-THEN and IF-THEN-ELSE stat m nts and CASE statem nts.

3. Th method according to claim 1, wher in the step of combining comprises the steps of:

   determining if the placement information provided by the user satisfies a predetermined plurality of rules; and

   if the information does not satisfy the predetermined plurality of rules, displaying appropriate error messages.

4. The method according to claim 3, wherein the predetermined plurality of rules require that one of the two selected flow forms is placed inside the other of the two selected flow form or one of the two selected flow forms is placed in a head-to-tail relationship with the other of the two selected flow form, only downward flow of control is required within the flowchart, and no two lines connecting flow forms cross one another.

5. An apparatus for assembling a flowchart on a programmable computer display comprising:

   means for displaying on the screen a set of basic flow form icons, each icon comprising at least one polygon having one input line and one output line, each icon having one input line and one output line;

   user input means for enabling a user to select at least one of the basic flow form icons; and

   means for displaying the selected icon on a different area of the screen in a position designated by the user.

6. The apparatus according to claim 5, further comprising:

   user input means for inputting text to be associated with the at least one polygon in the at least one selected icon; and

   means for displaying the inputted text in an area adjacent the at least polygon in the at least one selected icon.

7. The apparatus according to claim 5, wherein the set of basic flow form icons comprises a statement box icon having one polygon comprising one input line and one output line.

8. The apparatus according to claim 5, wherein the set of basic flow form icons comprises a sequence icon comprising two polygons connected by a lin , th s quenc icon comprising on input into a first of th two polygons and one output coming from a second of th tw polygons.

9. Th apparatus according to claim 5, wherein the

set of basic flow form icons comprises an iteration form icon for indicating FOR and WHILE language constructs, the iteration form icon comprising:

a first six sided polygon comprising one input line;

a four sided polygon connected to the first six-sided polygon by a line; and

a second six sided polygon connected to the four sided polygon by a line and comprising an output line.

10. The apparatus according to claim 5, wherein the set of basic flow form icons comprises two alternation form icons,

a first alternation form icon for indicating an IF-THEN language construct having a diamond polygon comprising an input line and one output line, the output line having a false branch extending straight downward from the lowest corner of the diamond polygon and a true branch extending to the right of the diamond polygon and comprising a rectangular or square polygon having an output line connected to the false branch such that the first one of the boolean form icons comprises one input line and one output line;

a second alternation form icon for indicating an IF-THEN-ELSE language construct having a diamond polygon comprising one input line and one output line, said output line splitting to extend both to the right and left side of the diamond polygon, the right branch being the THEN path of the construct and the left branch being the ELSE path of the construct, each branch comprising a rectangular or square polygon having an output line which bends toward each other to meet and form a single output line of the second boolean form icon.

11. The apparatus according to claim 5, wherein the set of basic flow form icons comprises an alternation form icon for indicating a CASE language construct, the alternation form icon comprising a diamond polygon having one input line and one output line, said output line splitting into a plurality of branches corresponding in number to the number of possibilities in the CASE language construct, each branch comprising a rectangular or square polygon having an output lien which bends toward each other to meet and form a single output line of the alternation form icon.

12. The apparatus according to claim 5, wherein each icon comprises a rectangular or square polygon and the means for displaying the selected icon comprises:

means for combining two selected flow form icons;

means for determining when the position designated for displaying the selected icon by the user is correct according to predetermined rules; and

means for prohibiting operation of the means for displaying the selected icon responsive to a determination by the means for determining that the position designated is incorrect according to the predetermined rules.

13. The apparatus according to claim 5, wherein the input line at a top point of the icon appears on the display directly above the output line at a bottom point of the icon.

14. An apparatus for assembling a flowchart on a computer display comprising:

means for displaying in a first area of the display a predetermined plurality of flowchart icons, each icon comprising at least one statement box, an input line and an output line;

user input means for indicating selection of one of the plurality of flowchart icons;

user input means for indicating placement of the selected icon in a second area of the display;

means for determining whether the indicated placement of the selected flowchart icon satisfies predetermined rules, the rules allowing for placement of one icon inside a statement box of another previously selected icon to form a new flowchart icon, allowing for placement of one icon in a head-to-tail relationship with another icon, allowing only downward flow of control within the flowchart, and prohibiting crossing of any flow line within the flowchart; and

means for temporarily adding to the predetermined plurality of flowchart icons any new flowchart icon created by the user input means for indicating placement.

15. The apparatus according to claim 14, further comprising:

user input means for inputting text to be associated with the at least one polygon in the at least one selected icon; and

means for displaying the inputted text in an area adjacent the at least polygon in the at least one selected icon.

16. The apparatus according to claim 14, wherein the predetermined plurality of flowchart icons comprises a statement box icon, a sequence icon comprising two connected statement boxes, an iteration form icon for forming FOR and WHILE statements, alternation form icons for forming IF-THEN and IF-THEN-ELSE statements and for forming CASE statements.

FIG. 1

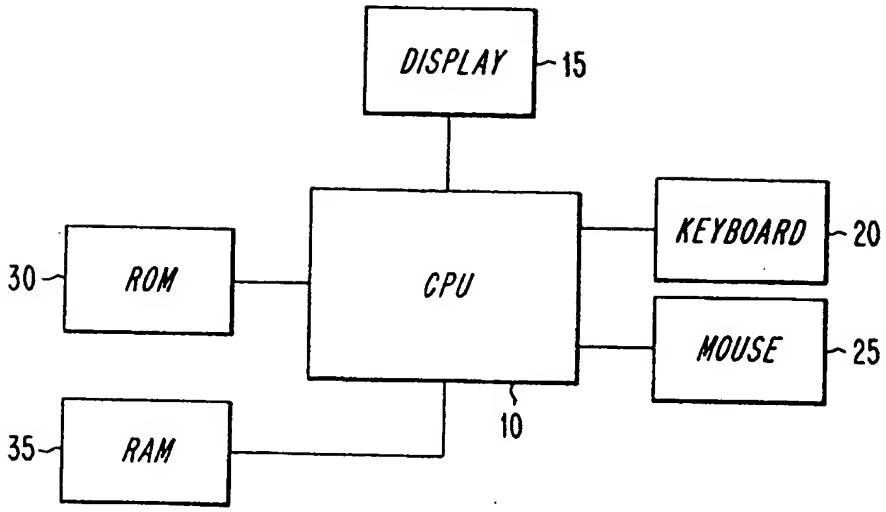


FIG. 2A

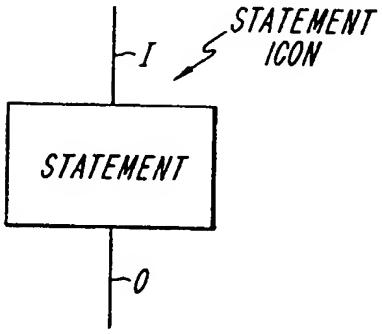


FIG. 2B

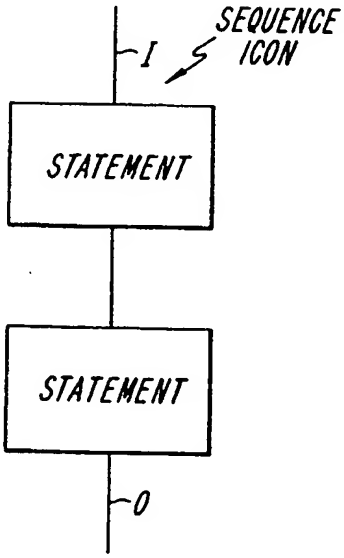


FIG. 2C

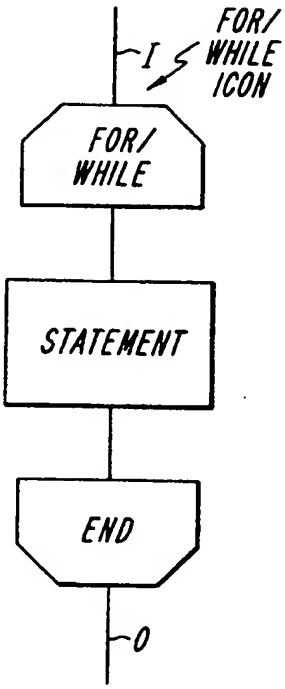


FIG. 2D

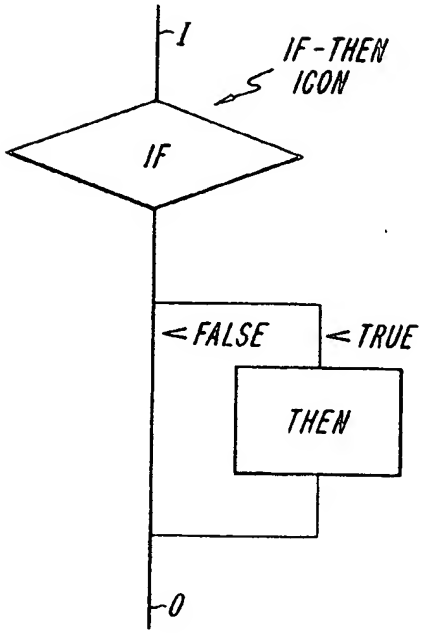


FIG. 2E

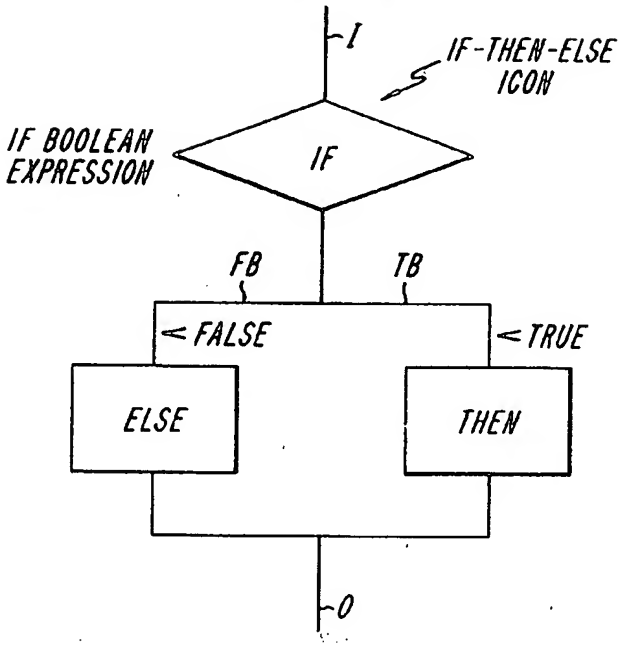


FIG. 2F

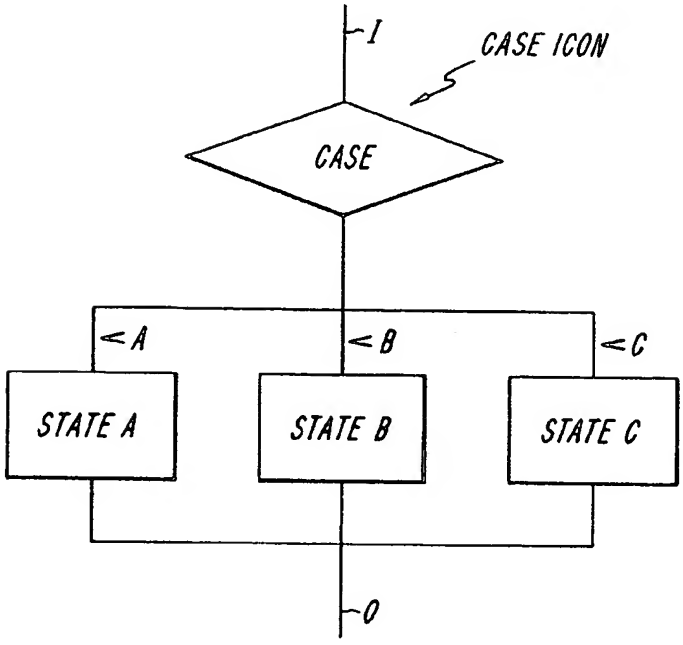


FIG. 3A

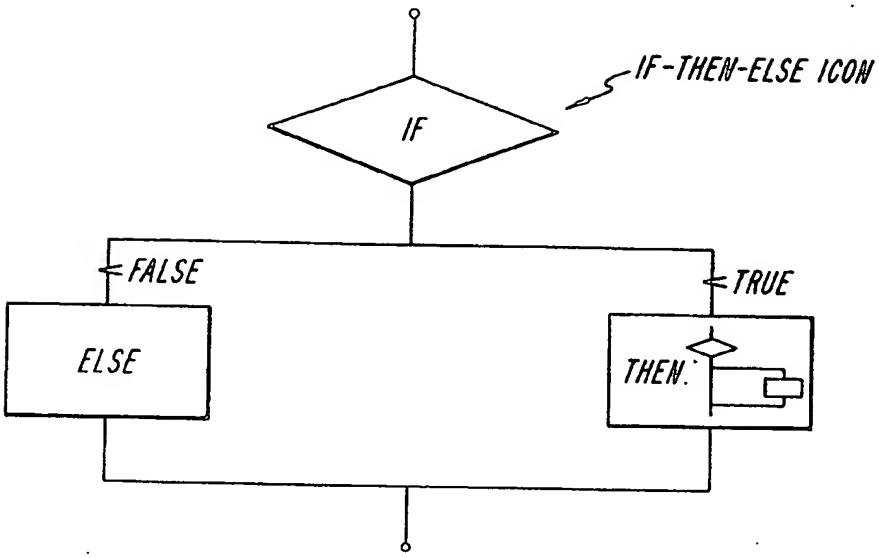
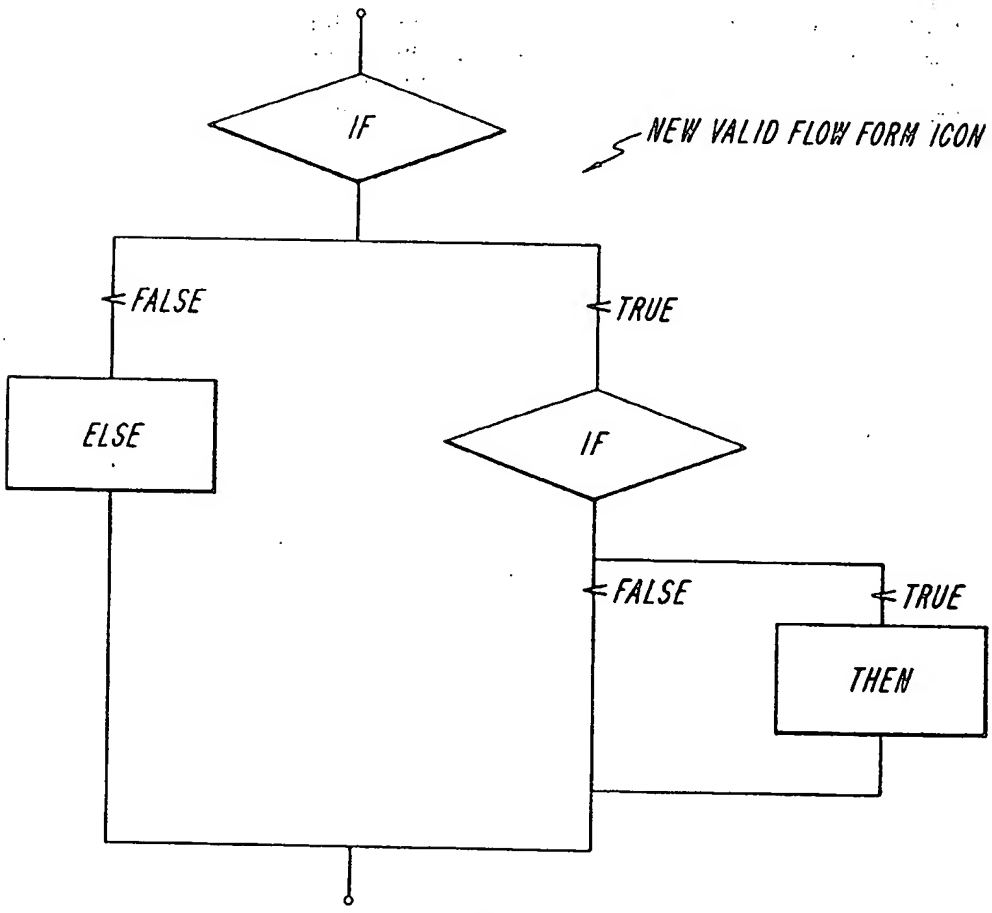

IF-THEN-ELSE ICON
IF
FALSE
TRUE
ELSE
THEN

FIG. 3B


IF
NEW VALID FLOW FORM ICON
FALSE
TRUE
ELSE
IF
FALSE
TRUE
THEN

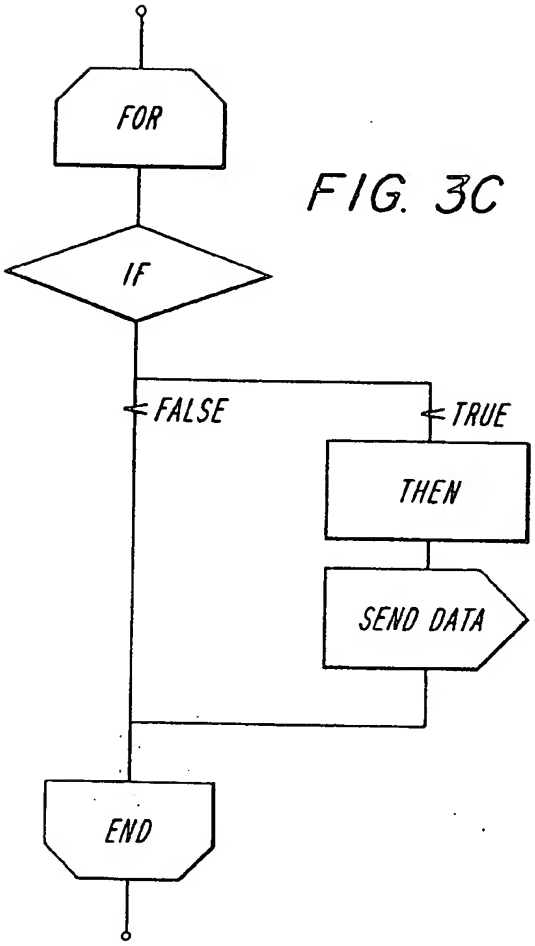

FOR
IF
FALSE
TRUE
THEN
SEND DATA
END

FIG. 3C

FIG. 3D

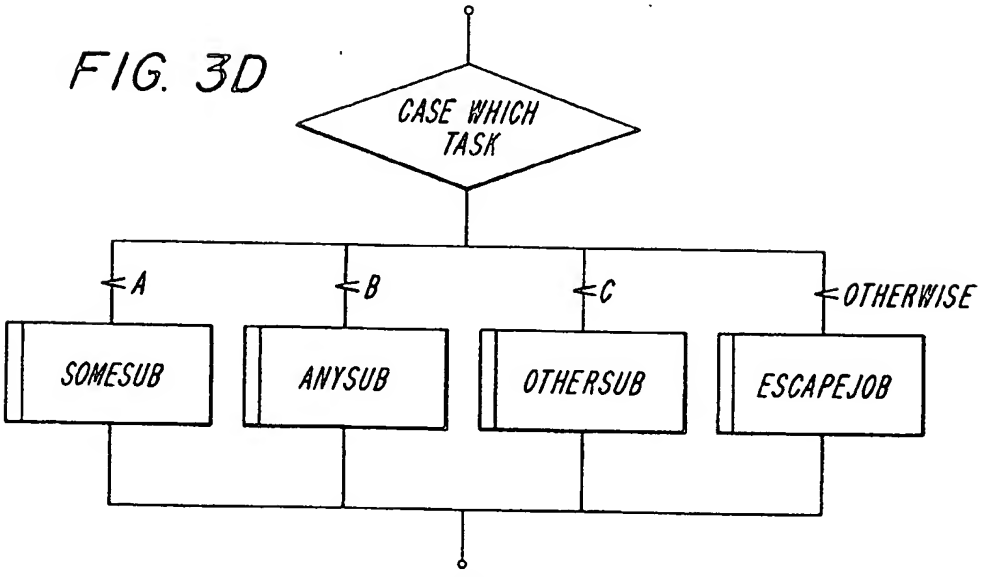

CASE WHICH TASK
A
B
C
OTHERWISE
SOMESUB
ANYSUB
OTHERSUB
ESCAPEJOB

FIG. 3E

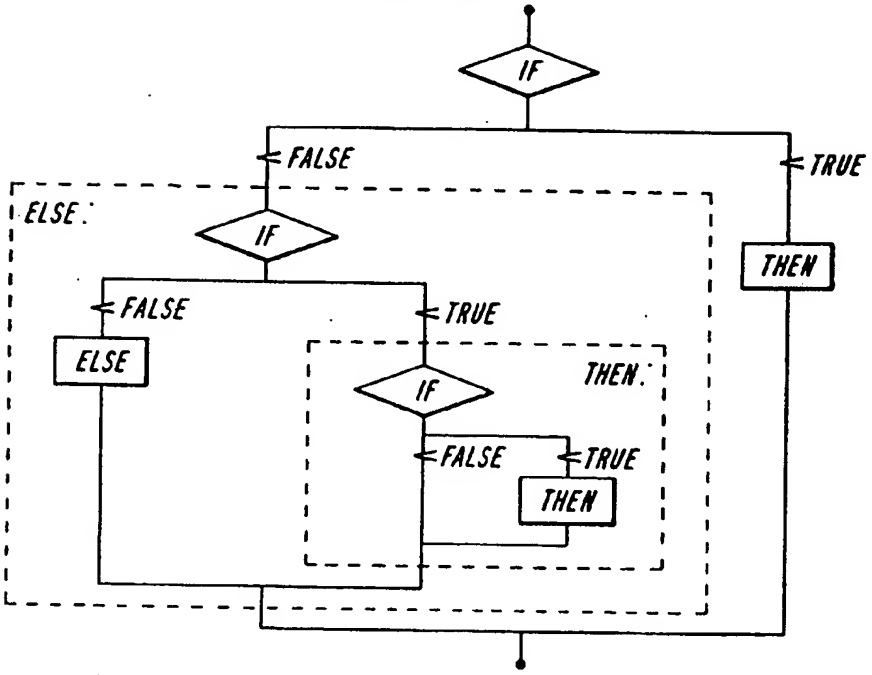
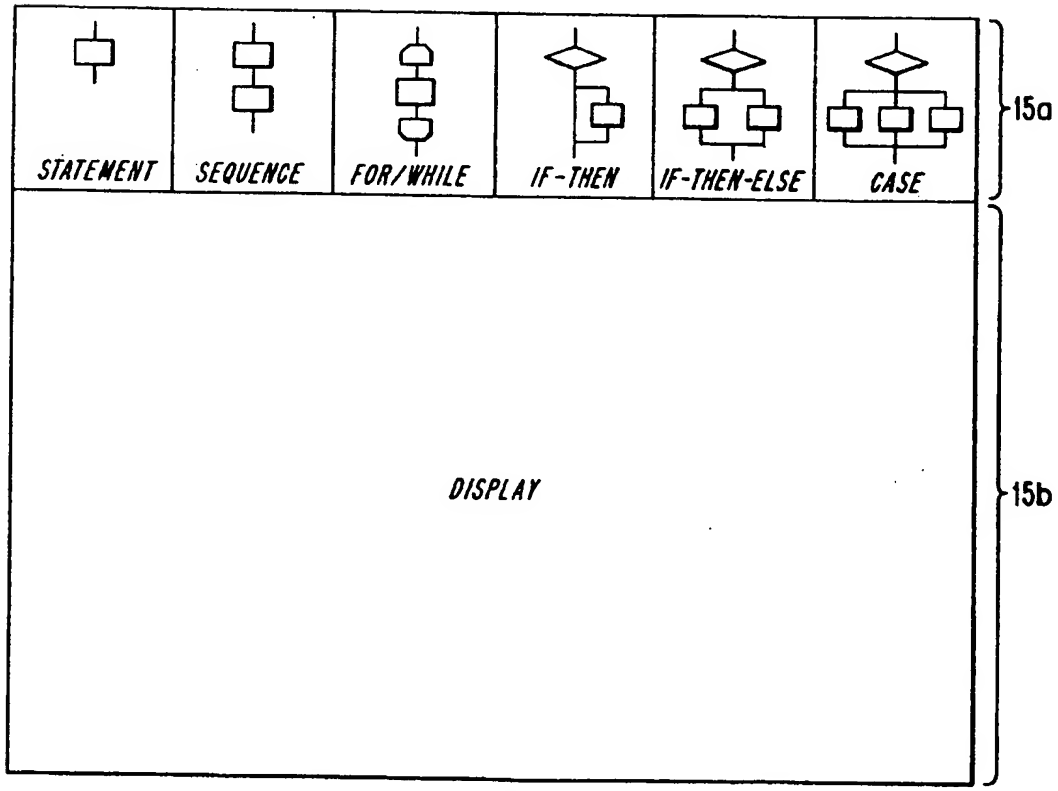

IF
FALSE
TRUE
ELSE
IF
FALSE
TRUE
ELSE
THEN
IF
FALSE
TRUE
THEN
THEN

FIG. 5


STATEMENT
SEQUENCE
FOR/WHILE
IF-THEN
IF-THEN-ELSE
CASE
15a
DISPLAY
15b

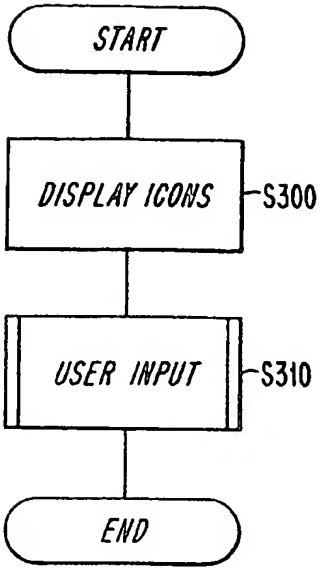
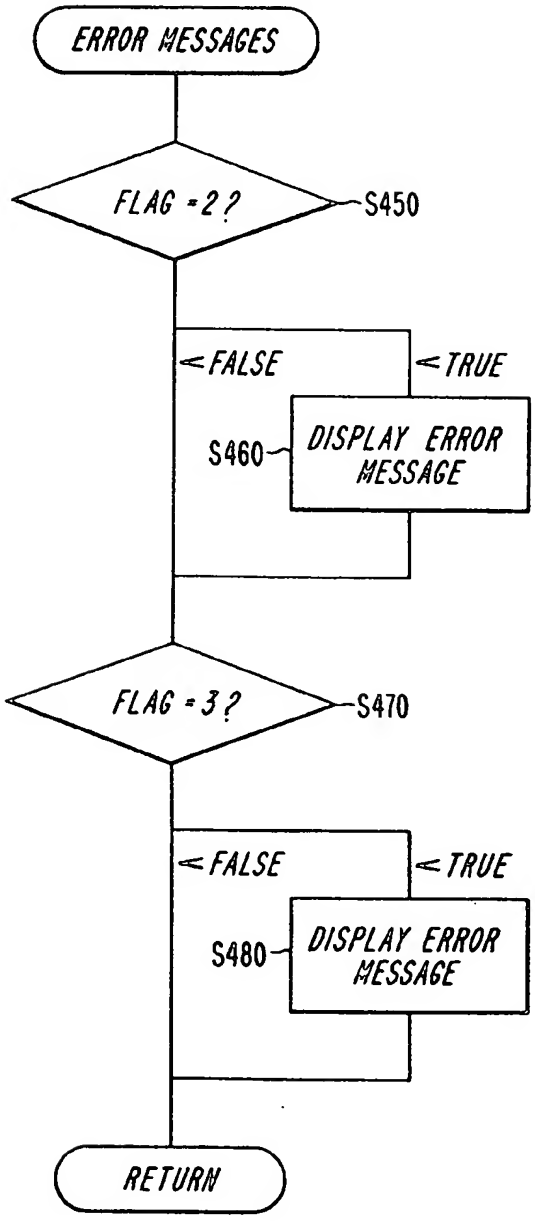

START
DISPLAY ICONS
S300
USER INPUT
S310
END

FIG. 4A

FIG. 4E


ERROR MESSAGES
FLAG = 2 ?
S450
FALSE
TRUE
S460
DISPLAY ERROR MESSAGE
FLAG = 3 ?
S470
FALSE
TRUE
S480
DISPLAY ERROR MESSAGE
RETURN

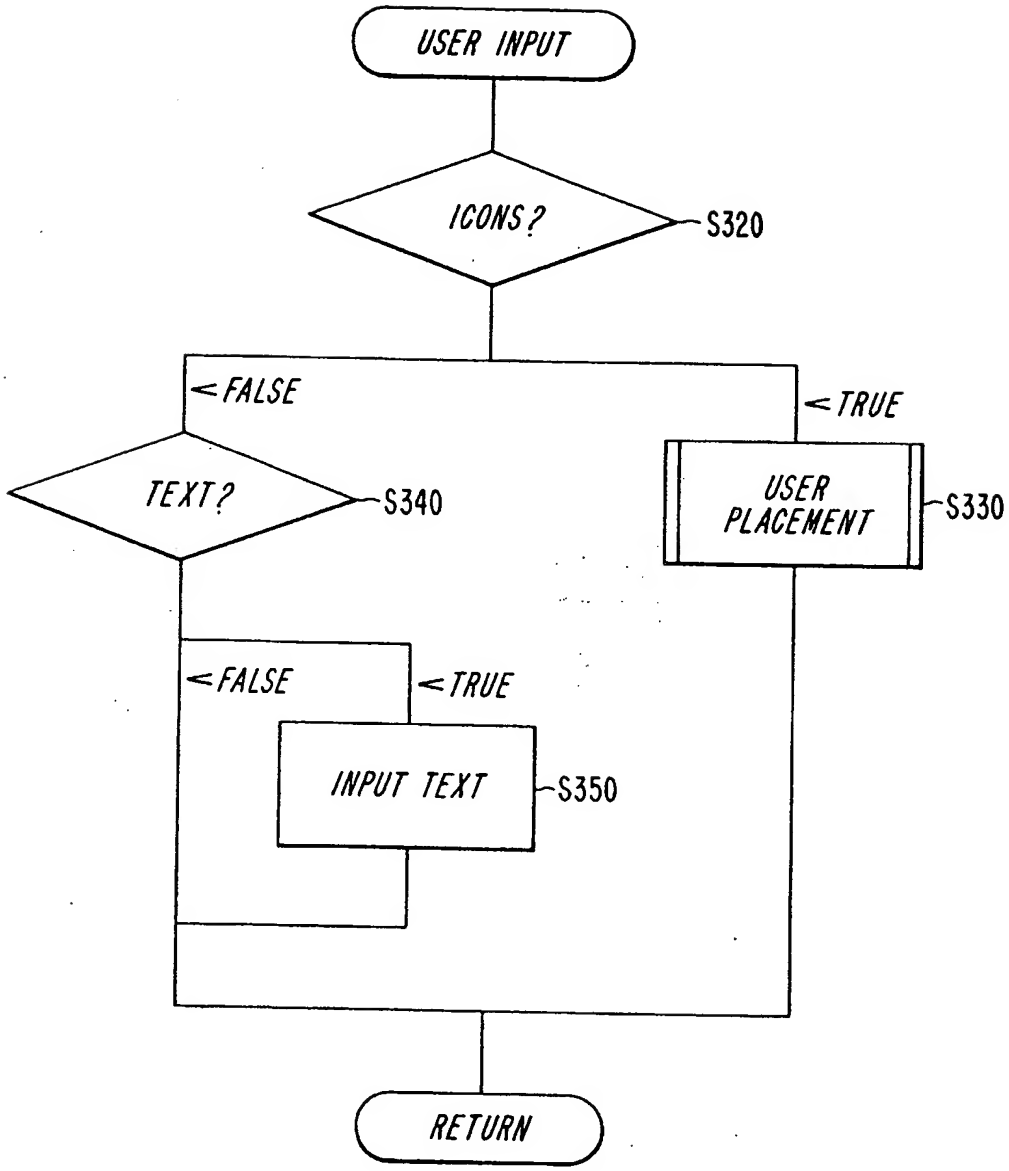
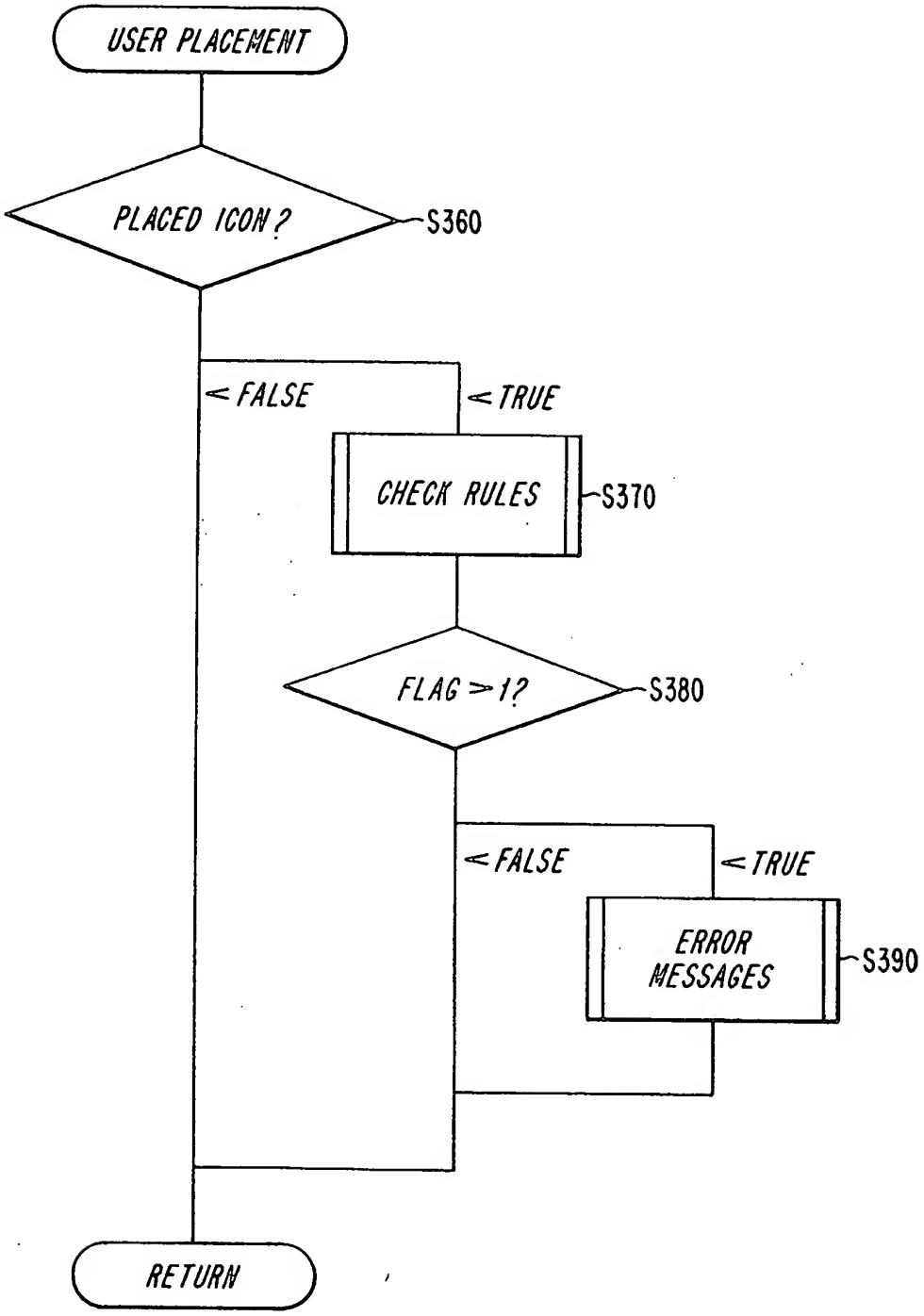

USER INPUT
ICONS?
S320
FALSE
TRUE
TEXT?
S340
USER PLACEMENT
S330
FALSE
TRUE
INPUT TEXT
S350
RETURN

FIG. 4B

FIG. 4C


USER PLACEMENT
PLACED ICON?
S360
FALSE
TRUE
CHECK RULES
S370
FLAG > 1?
S380
FALSE
TRUE
ERROR MESSAGES
S390
RETURN

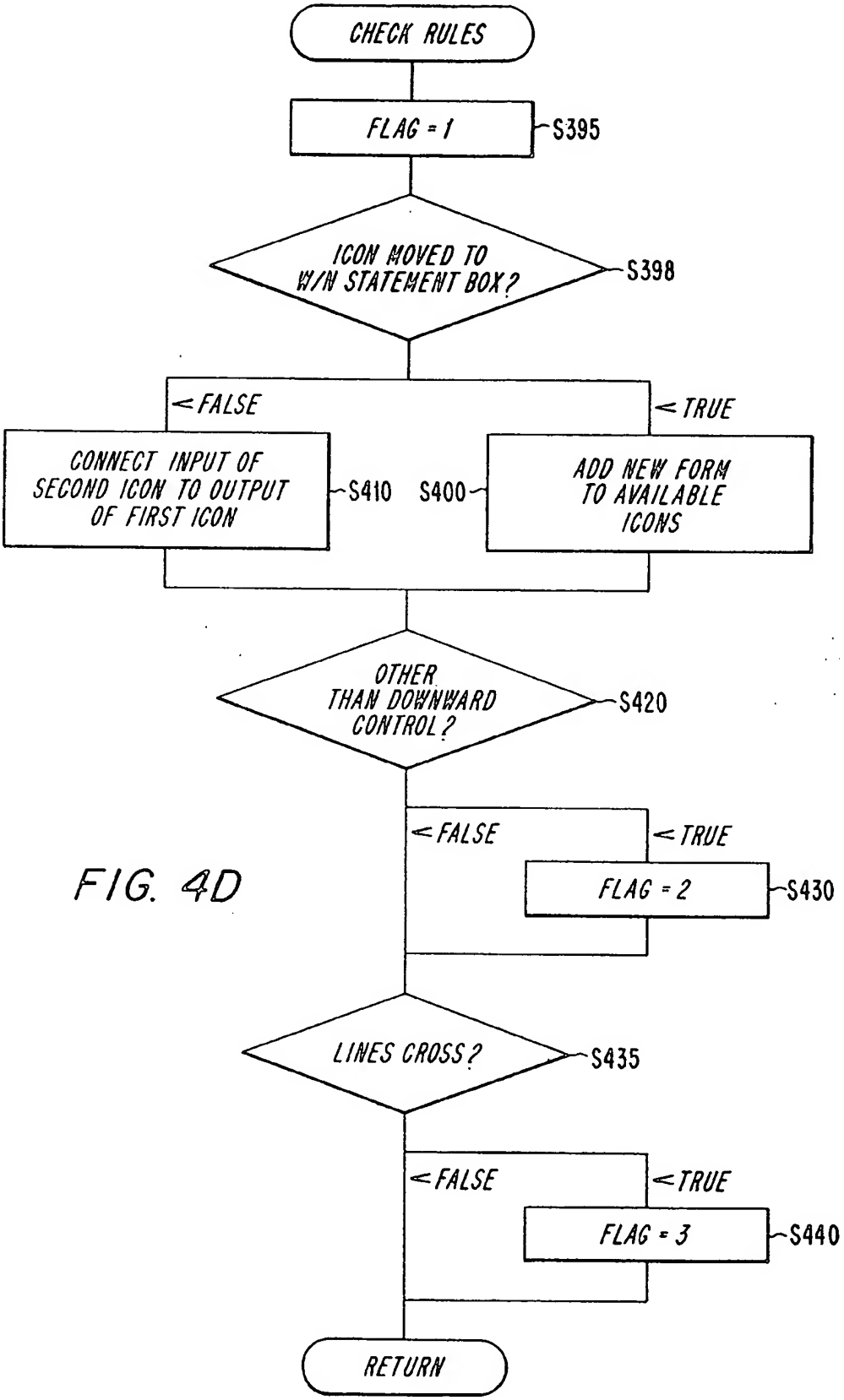

CHECK RULES
FLAG = 1
S395
ICON MOVED TO W/N STATEMENT BOX?
S398
FALSE
TRUE
CONNECT INPUT OF SECOND ICON TO OUTPUT OF FIRST ICON
S410
S400
ADD NEW FORM TO AVAILABLE ICONS
OTHER THAN DOWNWARD CONTROL?
S420
FALSE
TRUE
FLAG = 2
S430
LINES CROSS?
S435
FALSE
TRUE
FLAG = 3
S440
RETURN

FIG. 4D